The Round Table Model:
A Web-Oriented, Agent-Based Approach To Decision-Support Applications

Kym Jason Pohl
CAD Research Center
Cal Poly, San Luis Obispo, CA 93407

Abstract

Not unlike King Arthur relying on the infamous Round Table as the setting for consultation with his most trusted experts, agent-based, decision-support systems provide human decision makers with a means of solving complex problems through collaboration with collections of both human and computer-based expert agents. The Round Table Framework provides a formalized architecture together with a set of development and execution tools which can be utilized to design, develop, and execute agent-based, decision-support applications. Based on a three-tier architecture, Round Table incorporates forefront technologies including distributed-object servers, inference engines, and web-based presentation to provide a framework for collaborative, agent-based decision making systems.

Keywords

decision-support, computer-based decision making, artificial intelligence, agent, web-based computing, distributed systems, object-oriented DBMS, distributed object server, CORBA

Introduction

For King Arthur, it is a time of consultation and intense decision making. With much concern, King Arthur summons the knights of the Round Table, his most trusted experts, to take their place with him around the infamous *Round Table*. This is the setting they have chosen time and time again to discuss and plan the destiny of the great kingdom known as Camelot. In modern-day society, the need for an effective means of engaging in collaborative decision making is more prevalent than ever. With the development of newer, agent-based technologies, this need is beginning to be successfully addressed.

Object-Based Representation

Throughout the past decade the CAD Research Center (CADRC) at Cal Poly, San Luis Obispo, California has been intricately involved in the design and development of agent-based, decision-support systems from a practical standpoint (Pohl et al. 1997). As a result of these efforts, the CADRC has developed a manifesto of sorts describing a collection of criteria fundamental to the development of agent-based, decision-support systems (Pohl 1997).

First and foremost on this list is the need for an object-based representation of information. Information processed within the system must be described as objects having attributes, behavior, and relationships to other objects. Collectively, these descriptions form an application's information object model (Fowler and Scott 1997). This requirement not only applies to the modeling of information but at times is even portrayed in the representation of the agents themselves. It has been the experience of the CADRC that without such an objectified representation, where critical informational relationships can be captured, determination of information meaning and implication becomes extremely difficult if not impossible.

After numerous implementations it became clear to the members of the CADRC that to take full advantage of such objectified representation, a supportive framework needed to be established. A framework which centered around objects. Thus, the rational for *Round Table*.

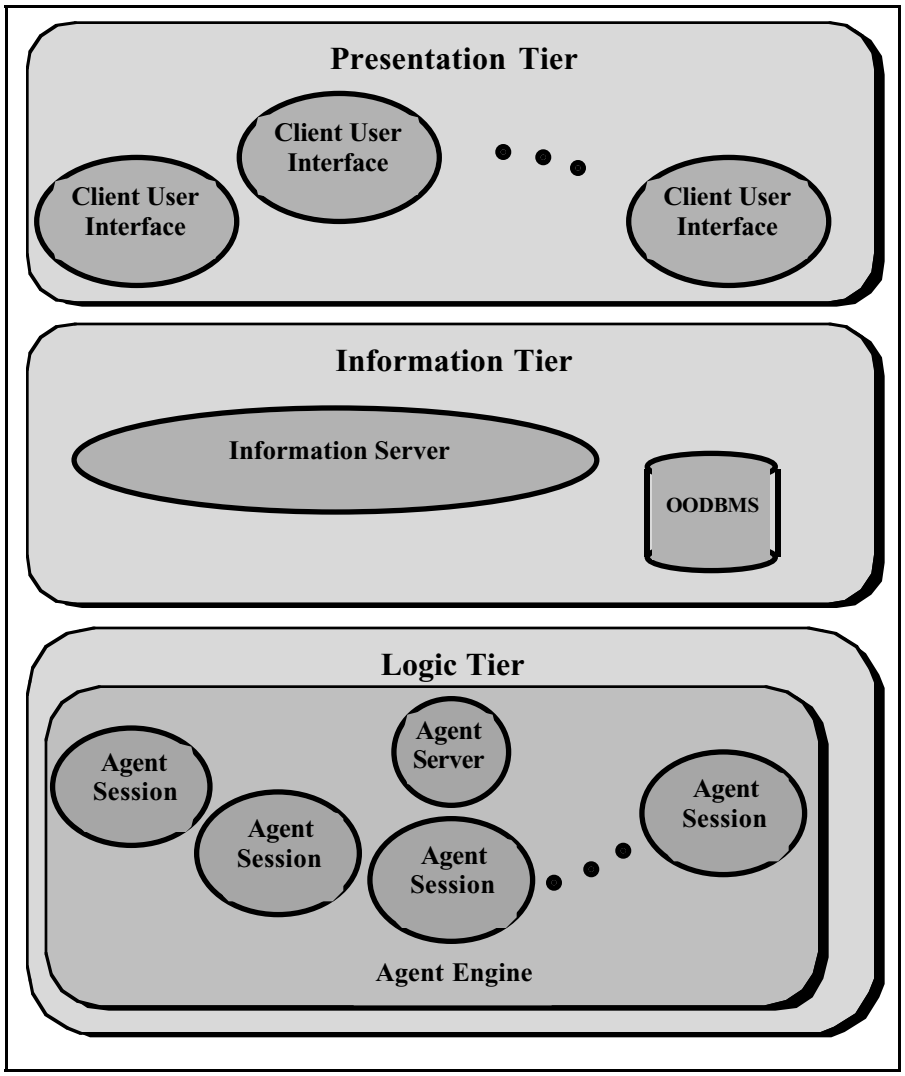

Figure 1 - Round Table Architecture

The Round Table Framework

The Round Table framework exists as an architecture, together with a set of development and execution tools which can be used to design, implement, and execute web-oriented, agent-based, decision-support applications.

The Round Table model is based on a three-tier architecture making clear and distinct separations between information, logic, and presentation (Gray et al. 1997). These tiers are represented by the three major components comprising the Round Table model; the Information Server (information tier), the Agent Engine (logic tier), and the Client User Interface (presentation tier) (Figure 1). Each of these components functions in an integrated fashion to form a comprehensive agent-based decision-support execution framework. This framework allows multiple human decision makers to solve complex problems in a collaborative fashion obtaining decision-support assistance from a collection of heterogeneous on-line agents.

Information Server

Core to the Round Table Framework is the Information Server (IS). Conceptually, the IS represents a library of objectified information which clients utilize to both obtain and contribute knowledge. The only difference is that clients can obtain this information in, not only a *pull* fashion, but can also have the IS *push* them information on a subscription basis. Physically, the IS exists as a distributed object server based on the Common Object Request Broker Architecture (CORBA) (Mowbray and Zahavi 1995).

Being the basis for the IS, distributed object servers are designed to service client requests for information. The knowledge of exactly where the information resides and how it can be retrieved is completely encapsulated inside the object server. This means that clients need not be concerned with who has what information and in what form that information exists. This feature becomes instrumental in providing an environment where collaborative application components operate in a de-coupled manner via the IS.

Regardless of the information's native representation, distributed object servers can be used to present information to clients in the form of objects. However, this does not discount the need for information to be modeled as high-level objects in its native form portraying behavior and conveying relationships. While on the surface this representational morphing capability of object servers seems promising, in practice this feature proves to be quite misleading. If the information is not represented at a high level upon its conception , such objectification amounts to little more than wrapping data in communicable object shells. These shells fail to convey any additional insight into the meaning or implication of the information than was present to begin with in its original form. Although in the future there may be potential for successful research efforts in this area, at present, unless information is originally modeled as objects, knowledge-oriented applications prove to gain little from this distributed object server feature.

However, applications that do, in fact, model information as high-level objects stand to gain considerably from employing distributed object servers. Distributed object servers preserve purely objectified representations of information as it moves throughout the system. This is due to the fact that the internal mechanisms of distributed object servers process information as objects themselves.

The Round Table model takes full advantage of these object-oriented facilities by integrating an Object-Oriented DBMS (Bancilhon et al. 1992) into its information environment. The OODBMS is the facility that the IS uses to store the application's objects. Employing an OODBMS to store the information objects has two significant advantages.

First, an OODBMS retains the object-oriented representational nature of the information as it exists in its persistent form. Whenever there is representational degradation there is potential for loss of informational content and meaning. By utilizing both transport and storage facilities which are capable of processing and manipulating information as objects, there is no degradation of representation as information flows throughout the application environment.

The second advantage of employing an OODBMS relates to the manner in which IS clients request information. Whether mining for information or posting a standing subscription, clients formulate their information requests in terms of objects. More specifically, in terms of object attributes and object relationships. These queries can range from simple existence criteria to the more complex incorporating both logical and relational operators. Foe example, one such query may request all InfoTech employees with a salary of more then $40,000. In this example, the client is essentially *pulling* information out of the IS. The operands of the query are each specified in terms of the application's object model.

Another method in which information can be obtained from the IS deals with the notion of subscription. Clients can dynamically register standing subscriptions with the IS which are again described in terms of the application's object model. For example, a client may requesting to be notified whenever InfoTech hires a new employee. Once registered, this condition is continually monitored by the IS. When satisfied, the IS essentially *pushes* the query results to whomever has indicated an interest (i.e., registered an appropriate subscription). The alternative to this subscription mechanism would be to have interested clients perform the same query on an iterative basis until such a condition occurs. Each unsatisfied query may potentially decrease resources (i.e., computing cycles) available to other application components and would essentially prove to be a waste of time. If a client takes a more conservative approach where the repeated query is made on a less frequent basis, the client risks being out of date with the current state of affairs until the next iteration is performed. With this in mind, the incorporation of a *push information*

to interested clients mechanism becomes essential in providing decision-support applications with an efficient, up-to-date operating environment.**Agent Engine**

The Agent Engine represents the logic-tier of Round Table's underlying three-tier architecture. Existing as a client to the IS the Agent Engine is capable of both obtaining and injecting information. Architecturally, the Agent Engine consists of an agent server capable of serving collections of agents (Figure 1). These collections, or Agent Sessions, exist as self-contained, self-managing agent communities capable of interacting with the IS to both acquire and inject information. For the most part, the exact nature of agents and collaborative model employed is left to the application specification. However, regardless of the types of agents contained in an Agent Session, agent activity is triggered by changes in application information. This information may take the form of global objects managed by the IS or local objects utilized in agent collaboration which are managed by the Agent Session itself. Regardless of whether agents are interacting with the IS or each other, interaction takes place in terms of objects. This again illustrates the degree to which an object representation is preserved as information is processed throughout the application environment.

Agent Session Configuration

Breaking agent analysis into heterogeneous collections of agents allows for a number of interesting configurations. These configurations determine the size, number, and individual scope of the agent sessions. While a wide variety of Agent Session configurations exist, the CADRC has found considerable success in formulating this configuration based on two primary criteria.

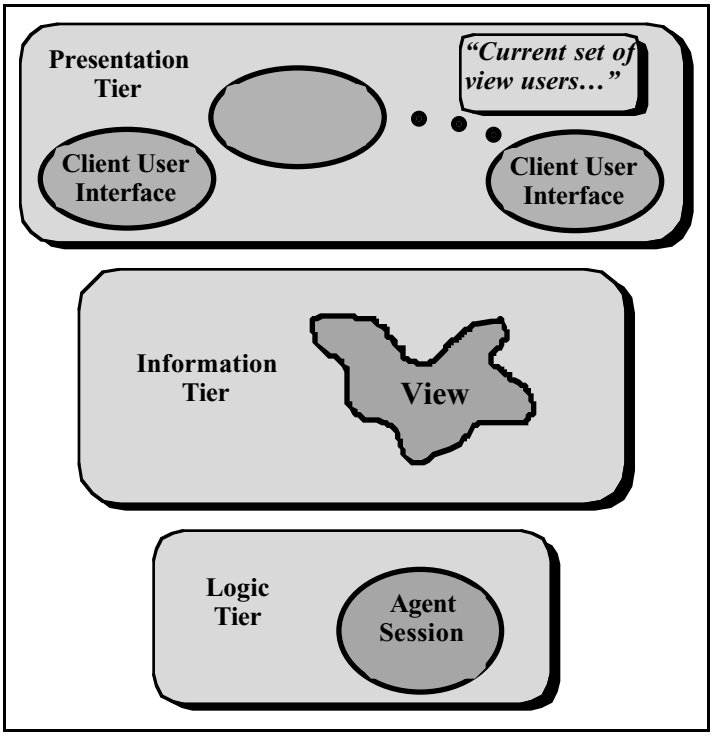

Figure 2 - Multiple users can interact with a view which in turn is analyzed by a single Agent Session

The first criterion introduces the notion of a *view*. A view is a conceptual perspective of reality. In other words, a view can be thought of as a single investigation into solving a problem whether it be based on fact or speculation. For example, a view may describe events and information relating to what is actually occurring in reality. Yet, another view may describe an alternative or desired reality. An illustration of this approach can be observed in the Integrated Marine Multi-Agent Command and Control System (IMMACCS) developed by the CADRC for the Marine Corps. IMMACCS uses a single view to represent the information and events occurring in the battlespace. In a similar manner, IMMACCS employs any number of additional views to represent hypothetical investigations to determine suitable strategies for dealing with potential events or circumstances. Regardless of use, however, there is a one-to-one correspondence between a conceptual view and an Agent Session (Figure 2). This means that independent of exactly which version of reality a view represents, there exists a dedicated Agent

Session providing users of that view with agent-based analysis and decision-support. Each agent of a particular Agent Session deals only with the view associated with its Agent Session. Organizing information analysis in this manner allows for an efficient and effective means of distinguishing activities relating to one view from activities pertaining to another. Unless prompted by user intervention, each set of information is completely separate from the other.

The second configuration criterion employed by the CADRC determines the quantity and nature of agents contained in an Agent Session at any point in time. As mentioned above, the decision-support applications developed by the CADRC utilize a variety of agent types. Three of these agent types include Domain Agents, Object Agents, and Mediator Agents (Pohl 1995). Recall that service-oriented Domain Agents embody expertise in various application-relevant domains (i.e., structural systems and thermal dynamics for architectural design, tidal dynamics and trim and stability for ship stow planning, etc.). The collection of Domain Agents populating an Agent Session at any point in time determines the variety of domain specific perspectives and analytical depth available during analysis of the associated view. Under the configuration scheme utilized by the CADRC, users can add or remove these domain perspectives in a dynamic fashion over time.

Object Agents, on-the-other-hand extend the notion of high-level informational representation by essentially agentifying information through empowering information objects with the ability to act on their own behalf. This agentification of information into Object Agents can be initiated by both human users or other agents as needed.

In an attempt to resolve conflicts arising between collaborating agents, Mediation Agents may be employed as third party mediators. It is the goal of these mediators to bring about consensus among agents that have reached an impasse.

Under the Round Table model each of these agent contingents is dynamically configurable by both the user(s) in addition to the system itself. This approach to Agent Session configuration promotes the notion of offering assistance in the form of dynamically configurable tools rather than predefined solutions (Pohl 1997).

Agent Session Architecture

Architecturally, an Agent Session consists of several components including the Semantic Network and Semantic Network Manager, Session Manager, Inference Engine, and Agent Manager (Figure 3). These components operate in an integrated fashion to maintain a current information connection between the agents residing in the Agent Session and the associated view described in the IS.

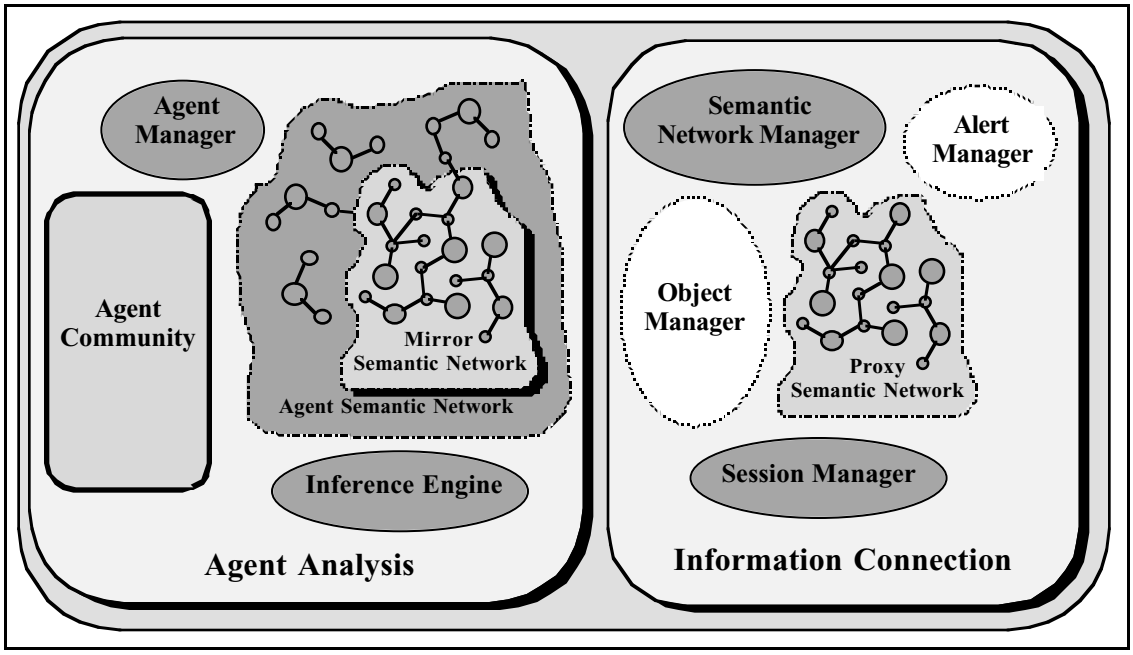

Figure 3 - Agent Session Architecture

Semantic Network

The Semantic Network consists of a collection of two sets of application specific information objects. The first set is used for local collaboration among agents. Depending on the specific collaborative model employed, agents may use this local Semantic Network to propose recommendations to each other or request various services. This information is produced and modified by the agents and remains local to the Agent Session. The second set of information is a sort of duplicate, mirror image of the view information stored in the IS. In actuality, this information exists as a collection of object-based interfaces allowing access to view information stored in the IS. Such interfaces are directly related to the application's information object model. In other words, these interfaces, or proxies (Mowbray and Zahavi 1995), are represented in terms of the objects described in the

information object model. Through these interfaces, IS clients have the ability to access and modify objects contained in the IS as though they are local to the client's environment. All communication between the object interfaces and their remote object counterparts is encapsulated and managed by the IS and completely transparent to the clients. This is a fundamental feature of distributed object servers on which the IS is based (Orfali et al. 1996).

Semantic Network Manager

As the primary manager of the two sets of information described above, the Semantic Network (SN) Manager focuses the majority of its efforts on the management of the bi-directional propagation of information between IS proxies and an equivalent representation understandable by the Inference Engine. Such propagation is accomplished through employing an Object Manager. The purpose of this manager is to essentially maintain mappings between the IS proxies and their corresponding Inference Engine counterparts. The necessity of this mapping reveals a limitation inherent in most distributed object server and inference engine facilities. Most facilities supporting one of these two services require control over either the way client information is represented or the manner in which it is generated. This is due to the fact that both facilities require specific behavior to be present in each object they process. Examples of such facilities include IONA's ORBIX distributed object server (IONA 1996) and NASA's CLIPS inference engine (NASA 1992). Both of these facilities suffer from this limitation. Nonetheless, this dilemma can be solved through the use of an intermediate object manager which maintains mappings between the two sets of objects.

An additional responsibility of the SN Manager deals with the subscriptions, or interests held on behalf of the agent community. That is, the SN Manager is responsible for maintaining the registration of a dynamically changing set of information interests held on behalf of the Agent Session agents. In addition, the SN Manager is responsible for processing notification(s) when these interests are subsequently satisfied. Such processing includes the propagation of information changes to the agent community which may in turn trigger agent activity. To perform these two interest-related tasks the SN Manager employs the services of the Alert Manager. The Alert Manager exists as an interface to the IS subscription facility and is available to any IS client wishing to maintain a set of information interests. Employment of the Alert Manager by subscribers has two distinct advantages. First, IS clients are effectively de-coupled from the specifics of the IS subscription interface. This allows the same application client to be compatible with a variety of object server implementations. Second, the Alert Manager interface allows subscribers to effectively decompose themselves into a dynamic collection of thread-based interest clients (Lewis and Berg 1996). That is, the Alert Manager extends the monolithic one-to-one relationship between the IS Server and an IS client into one which supports a one-to-many relationship. Such decomposition of functionally related behavior into light-weight processes promotes the concepts of multi-processing in conjunction with resource conservation.

Inference Engine

The Inference Engine provides the link between changes occurring in the Semantic Network and agent activation. Recall that agent activation can occur when a change in the Semantic Network is of interest to a particular agent. In such a case, the Inference Engine, having knowledge of specific agent interests in addition to changes occurring in the Semantic Network is responsible for activating, or scheduling the action(s) the agent wishes to execute. This activation list forms the basis for the Agent Manager to determine which agent actions to execute on behalf of the currently scheduled agent.

Agent Manager

The Agent Manager is responsible for the management of the agent community housed in an Agent Session. This management includes the instantiation and destruction of agents as they are dynamically allocated and deallocated to and from the agent community. In addition, the Agent Manager is responsible for managing the distribution of execution cycles allowing each agent to perform actions. Dispersement of execution cycles occurs in a round-robin fashion allowing agent analysis to be evenly distributed among relevant agents. Whether or not an agent utilizes its allotted cycles depends on whether it has any tasks or actions to perform.

Session Manager

As the overall manager of the Agent Session environment the Session Manager has two main responsibilities. The first of these responsibilities focuses on the initialization of each of the other Agent Session components upon creation. When an Agent Session is created as a response to the creation of a view, the Session Manager is the first component to be activated. Once initialized, the Session Manager activates the SN Manager and Inference Engine. Continuing its efforts, the Session Manager then activates the Agent Manager. Upon startup, the Agent Manager initializes itself by allocating an appropriate initial set of agents. Depending on the application specifics, these agents may in turn perform a series of initial queries and subscriptions which will eventually propagate to the IS via the SN Manager.

Client User Interface

Representing the third and final tier of the three-tier architecture employed by Round Table the Client User Interface (CUI) exists as a web-based application which can operate in a light-weight computing environment. The CUI essentially provides human users with a means of viewing and manipulating the information and analysis provided by the other two tiers of the agent-based, decision-support application. Understanding the importance of data presentation, the CUI presents the user with this information and analysis in a robust and graphical manner.

As clients of the IS, CUI users have the ability to interact with each other in a collaborative fashion. That is, by virtue of either injecting or obtaining information from the IS, CUI users working on the same view have the potential of exchanging design information in a collaborative manner. This type of information exchange occurs regardless of whether the relevant view represents the main design effort or exists as a localized solution attempt explored by a subset of users. All information and analysis remains localized within its particular view unless explicitly copied into another view as a user initiated action. In this manner, no informational or analytical collisions occur between conceptual views without the potential for user-based supervision and subsequent reconciliation.

Future Research

As a further formalization of the Round Table approach to agent-based, decision-support applications, the creation of a robust collection of design and development tools is planned within the near future. These tools promise to combine the roles of application designer and application developer into a single effort. Decision-support applications can be designed and developed through a series of high-level models describing information structure and analytical logic. High-level classes can be identified through a series of Unified Modeling Language (UML) class diagrams forming a comprehensive information object model. This model essentially describes the application specific design and problem space as a collection of high-level objects complete with attributes and relationships. This is the same high-level description of application information which was identified earlier as being crucial to agent-based, decision-support applications.

By the same token, much of the analytical reasoning applied to this information can be described in terms of a methodology suitable for representing logic. The methodology intended to be employed by this set of design and development tools attempts to represent logic as a series of rules (Hayes-Roth et al. 1983). Each of these rules identifies both a condition and a corresponding action to take upon the satisfaction of that condition. This is where the advantages of using a high-level, object-based representation again become apparent. Both the condition and action components of these rules can be described in terms of the application's information object model. That is, conditions can be represented as a series of references to object attributes strung together with logical and relational operators. The corresponding action is itself described in terms of the object model. When the informational state described in the condition section of the rule occurs, the corresponding action component will modify or produce information thus creating an entirely new informational state. This new state may in turn trigger other rules to execute in a similar fashion. Although not all logic can be represented in this manner, it is the authors expectation that this approach can be applied to a significant portion of analytical reasoning found in decision-support applications.

Once both the information and portions of the logic are described as high-level design models, much of the decision-support application can be automatically generated. The object model can be used as a basis for automatically generating any object-specific behavior required by the various Round Table components outlined in this paper including the IS, Agent Engine, and CUI. In a similar manner, the logic model can be used to automatically generate the condition and action components of rules which essentially form a significant portion of the application's agent communities. Such automatic generation is possible because the information required to implement the application-specific portions of these Round Table components is present in a concise and unambiguous fashion within these two design perspectives.

By elevating the vast majority of agent-based, decision-support application development to the level of conceptual design, such applications can be developed, maintained, and modified in a considerably more efficient and proficient manner as compared to manual development. Further, this approach essentially eliminates the loss of intent which often occurs as application development moves from the designers to the program developers. Utilizing the Round Table model together with its design and development tools, these roles become synonymous.

References

Bancilhon F, C Delobel and P Kanellakis (eds.); "*Building an Object-Oriented Database Systems*"; Morgan Kaufman, San Mateo, CA, 1992.

Fowler M and K Scott; "*UML Distilled: Applying the Standard Object Modeling Language*"; Addison-Wesley, Reading, Massachusetts, 1997.

Gray S and R Lievano; "*Microsoft Transaction Server 2.0*"; SAMS Publishing, Indianapolis, Indiana, 1997.

Hayes-Roth F, D Waterman and D Lenat (eds.); "*Building Expert Systems*"; Addison-Wesley, Reading, Massachusetts, 1983.

IONA; "*Orbix Web: Programming Guide*"; IONA Technologies Ltd., Dublin, Ireland, 1996.

Lewis B and D J Berg; "*Threads Primer: A Guide to Multithreaded Programming*"; SunSoft Press; Mountain View, CA, 1996.

Mowbray T and R Zahavi; "*The Essential CORBA: Systems Integration Using Distributed Objects*"; John Wiley and Sons, Inc., New York, New York, CA, 1995.

NASA; "*CLIPS 6.0 Reference Manual*"; Software Technologies Branch, Lyndon B Space Center, Houston, Texas, 1992.

Orfali R, D Harkey and J Edwards; "*The Essential Distributed Objects Survival Guide*"; John Wiley and Sons, Inc., New York, New York, CA, 1996.

Penmetcha K, A Chapman and A Antelman; "*CIAT: Collaborative Infrastructure Assessment Tool*"; in Pohl J (ed.) Advances in Collaborative Design and Decision-Support Systems, focus symposium: International Conference on Systems Research, Informatics and Cybernetics, Baden-Baden, Germany, August 18-22, 1997 (pp. 83-90).

Pohl J, A Chapman, K Pohl, J Primrose and A Wozniak; "*Decision-Support Systems: Notions, Prototypes, and In-Use Applications*"; Technical Report, CADRU-11-97, CAD Research Center, Design Institute, College of Architecture and Environmental Design, Cal Poly, San Luis Obispo, CA, January, 1997.

Pohl J; "*Human-Computer Partnership in Decision-Support Systems: Some Design Guidelines*"; in Pohl J (ed.) Advances in Collaborative Design and Decision-Support

Systems, focus symposium: International Conference on Systems Research, Informatics and Cybernetics, Baden-Baden, Germany, August 18-22, 1997 (pp. 71-82).

Pohl K; "*KOALA: An Object-Agent Design System*"; in Pohl J (ed.) Advances in Cooperative Environmental Decision Systems, focus symposium: International Conference on Systems Research, Informatics and Cybernetics, Baden-Baden, Germany, August 14-18, 1995 (pp. 81-92).